SMART PATIENT HEALTH MONITORING SYSTEM

Submitted in the partial fulfillment for the requirements for the award of the degree of

Bachelor of Engineering

in

Electronics and Telecommunication Engineering

By

Samarth Sawant (B-836)
Sanskar Patil (B-840)
Rishikesh Patil (B-837)

Under the Supervision of

Prof. Kavita Rathi

Rajiv Gandhi Institute of Technology, Mumbai

(Affiliated to the University of Mumbai)

MGT
MANJARA CHARITABLE TRUST
RAJIV GANDHI INSTITUTE OF TECHNOLOGY, MUMBAI

(Permanently Affiliated to University of Mumbai)

Academic Year 2024-25

Certificate

This is to certify that the project entitled **SMART PATIENT HEALTH MONITORING SYSTEM** is a bona fide work of Samarth Sawant (Roll No. B-836), Sanskar Patil(B-840), and Rishikesh Patil (Roll No. B-837) under the supervision of Prof. Kavita Rathi. This project is submitted to the University of Mumbai in partial fulfillment of the requirement for the award of the degree of Bachelor of Engineering in Electronics and Telecommunication Engineering.

Prof. Kavita Rathi
Supervisor

Dr. Sanjay D. Deshmukh
Head of Department

Dr. Sanjay U. Bokade
Principal

Project 1 Report Approval for B.E.

This is to certify that the report titled **SMART PATIENT HEALTH MONITORING SYSTEM** has been successfully completed by:

- Samarth Sawant (Roll No. B-836)
- Sanskar Patil (Roll no. B-840)
- Rishikesh Patil (Roll No. B-837)

The project has been reviewed and approved by the undersigned for the partial fulfillment of the requirements for the award of the degree of Bachelor of Engineering in Electronics and Telecommunication Engineering.

Prof. Kavita Rathi
Project Guide

Dr. Sanjay D. Deshmukh
Head of Department

Dr. Sanjay U. Bokade
Principal

Declaration

We, the undersigned, declare that the project entitled **SMART PATIENT HEALTH MONITORING SYSTEM** is a bona fide work carried out by us for the partial fulfillment of the requirements for the degree of Bachelor of Engineering in Electronics and Telecommunication Engineering at Rajiv Gandhi Institute of Technology, Mumbai. The work presented in this report is original and has not been submitted earlier to any other university or institution for the award of any degree or diploma.

Samarth Sawant
Roll No. B-836

Sanskar Patil
Roll No. B-840

Rishikesh Patil
Roll No. B-837

Prof. Kavita Rathi
Project Guide

Abstract

This document presents a Smart Patient Health Monitoring System that uses intelligent technology to support patient care and maintenance in the Intensive Care Unit (ICU). Traditional ICU care systems often rely on manual assessments, which can be slow to identify significant changes in a patient's health. The integration of IoT sensors, data analytics, and communication technologies helps automate the monitoring process, providing real-time, continuous, and personalized care.

This system allows for:

- *Real-time monitoring of vital parameters like heart rate, SpO2, and temperature.*
- *Automated alerts sent to medical staff via mobile notifications when patient vitals exceed safe thresholds.*
- *Secure and scalable data storage using a cloud-based platform.*
- *Predictive analytics to identify potential health risks and enable timely intervention.*

The aim of this project is to improve the efficiency of monitoring systems in ICUs, reduce the burden on healthcare professionals, and ensure that patients receive timely intervention when required.

Contents

Chapter 1

Introduction

The intensive care unit (ICU) is a critical hospital environment where critically ill and lifethreatening patients are cared for. Patient care in the ICU is complex and requires con stant monitoring of vital parameters such as heart rate, blood pressure, blood oxygen level (SpO2), and body temperature. Traditionally, patient care in the ICU has relied heavily on bedside equipment and physician interventions. However, these methods can sometimes delay detection of emergencies or subtle changes in a patient's health, affecting patient outcomes. With the rise of the Internet of Things (IoT) and cloudbased technology, the healthcare industry has begun exploring new ways to increase the efficiency and reliability of patient care. Intelligent Patient Health Monitoring Systems Leverage these advances to address current limitations. The system is designed to collect realtime data from sensors attached to the patient, process it using cloud computing, and automatically alert doctors to detect defects during the exam. This approach ensures timely medical intervention and reduces the risk of death. Remote access. This continuous monitoring can also help reduce the workload of ICU staff, allowing them to focus on other important tasks while being alerted to crisis situations. The system's scalability makes it suitable for a variety of ICU settings, and multiple sens ors can be added or integrated with hospital data for seamless health management. We're making ICU care smarter and more efficient. Smart patient healthcare is not just a techno logical innovation, but also a step towards making healthcare more efficient. The system works to save lives and improve the quality of care by increasing the reliability, speed and accessibility of important patient information.

Chapter 2

Problem Statement and Objectives

2.1 Problem Statement

- Intensive care units (ICUs) play a vital role in saving lives, but current care systems have limitations that can impact patient outcomes. Routine critical care monitoring involves monitoring vital signs and relying on medical staff to detect abnormalities and take action quickly. When nurses and doctors are stressed or when monitoring equipment doesn't sound the alarm immediately, important events can be missed and treatment can be delayed. Additionally, the lack of remote access to patient information can be problematic because decisions can't be made immediately. Risk of delay
- Overcrowding of healthcare staff: As the number of patients increases, it becomes difficult for doctors to regularly attend to all their patients, which can lead to human error. Warning to delay medical intervention in critical situations. The system cannot use diagnostic tools to detect early signs of damage, reducing the risk of necessary care
- Manual Monitoring Limitations: Traditional ICU monitoring depends heavily on manual observation, increasing the risk of delays in detecting emergencies
- Overburdened Healthcare Staff: With increasing patient loads, healthcare personnel may struggle to monitor every patient continuously, leading to human errors.
- Lack of Real-Time Alerts: Conventional systems may not provide instant alerts, resulting in delayed medical interventions during critical situations
- No Remote Access: Patient data is often accessible only at the bedside, limiting doctors' ability to monitor patients remotely.

2.2 Objectives

- Real-Time Monitoring: Capture vital parameters such as heart rate, SpO2, blood pressure, and body temperature continuously using IoT sensors.
- Automated Alerts: Develop a notification system to instantly alert medical staff of abnormalities via mobile or web platforms
- Remote Access: Enable healthcare professionals to monitor patient data remotely for timely decision-making
- Predictive Analytics: Use data analytics to identify health patterns and predict potential risks for proactive care.
- Scalable System: Design a flexible system that allows integration with additional sensors or hospital systems based on requirements
- Data Security: Ensure patient privacy and data security through encryption and compliance with healthcare regulations

Chapter 3

Literature Survey

In recent years, significant progress has been made at the intersection of medicine and technology, with many studies investigating new ways to improve patient care and treatment. The following papers highlight important research in this area, highlighting many aspects of healthcare and technology.

1. Al Alkeem et al. (2017) introduced a new secure healthcare system utilizing the Cloud of Things (CoT). Their study emphasized the integration of cloud computing and IoT technologies to ensure secure and efficient health data management [**?**].

2. Kim et al. (2014) investigated the coexistence of ZigBee-based Wireless Body Area Networks (WBANs) and WiFi for health telemonitoring systems. Their research focused on addressing interference issues and optimizing network performance to facilitate reliable health data transmission [**?**].

3. Baig and Gholamhosseini (2013) provided an overview of smart health monitoring systems, emphasizing design principles and modeling techniques. Their study outlined various components and functionalities of these systems, highlighting their potential in improving healthcare outcomes [**?**].

4. Riazulislam et al. (2015) conducted a comprehensive survey on the Internet of Things (IoT) for healthcare. Their research explored various IoT applications, including remote patient monitoring, wearable devices, and healthcare analytics, underscoring the transformative impact of IoT in the healthcare domain [**?**].

Chapter 4

Methodology

4.1 System Design and Architecture

4.1.1 Patient-side System Architecture

The **Smart ICU** system is designed to monitor a patient's health status and provide real-time updates to healthcare providers through a web-based frontend. The system architecture consists of multiple components working in unison to collect, simulate, display, and manage the health data of the patient.

Components Used

- **Arduino Mega 2560:** Acts as the central processing unit in the system. It simulates various sensors like heart rate, body temperature, and other health metrics. It communicates with the sensors (real or simulated), collects data, and transmits it for processing or visualization.
- **16x2 LCD:** This display is used to show critical health information to the healthcare provider or patient on-site, such as heart rate, body temperature, and patient status. In the simulation, this is simulated using the frontend, as the LCD is replaced by dynamic displays (e.g., React UI).
- **Simulated Sensors:**
  - **Air Quality Sensor (e.g., MQ-135):** Measures the quality of air surrounding the patient, detecting pollutants such as CO2, alcohol, ammonia, and other gases. The sensor's output is simulated with varying air quality values to represent environmental conditions.
  - **Toxic Gas Sensor (e.g., MQ-7, MQ-9):** Detects the presence of toxic gases such as carbon monoxide (CO) and other harmful substances. In the simulation, the values are simulated to monitor the exposure to harmful gases.
  - **ECG Sensor (e.g., AD8232):** Monitors the electrical activity of the heart by measuring the ECG signal. In the simulation, the ECG values are generated to represent the patient's heart rhythm and condition.

  - **SpO2 Sensor (e.g., MAX30100):** Measures the oxygen saturation in the blood. The simulation mimics the real sensor data, with values ranging from 95-100
  - **Body Movement Sensor (e.g., MPU6050 Accelerometer):** Tracks the movement and orientation of the patient. The sensor data is simulated in the frontend to detect any significant body movements such as walking or changes in posture.
  - **Pressure Sensor (e.g., BMP180):** Measures the atmospheric pressure and temperature around the patient. In the simulation, the values are generated to mimic real-world atmospheric pressure data.
  - **LM35 Temperature Sensor:** Monitors the patient's body temperature. In the simulation, the temperature value is randomly generated to simulate real-time body temperature.
  - **Humidity Sensor (e.g., DHT11/DHT22):** Measures the relative humidity of the environment. The sensor value is simulated with varying levels of humidity, which may impact the patient's comfort and health.
  - **Alarm System:** The alarm system is used to notify healthcare providers when a sensor reading exceeds predefined thresholds, such as abnormal temperature, heart rate, or SpO2 levels. In the simulation, the alarm is triggered when critical values are detected, alerting the staff.

- **GPS Tracking:** Simulated in the frontend, this system allows for real-time tracking of the patient's movement. It uses the patient's location data (latitude and longitude) to update a Google Map and show the live location on the user interface.

How Each Part of the System Contributes to Monitoring the Patient

- **Sensors:** These components continuously collect data related to the patient's health. The Arduino Mega processes the sensor outputs and sends them to the cloud or the frontend for visualization. The sensors include the air quality sensor, toxic gas sensor, ECG sensor, SpO2 sensor, body movement sensor, pressure sensor, LM35 temperature sensor, and humidity sensor, all of which provide critical health and environmental data.

- **Simulation:** In the absence of real hardware, the sensors' data (e.g., temperature, heart rate, body movement, air quality) is simulated in the frontend. The simulated data mimics real sensor outputs, allowing healthcare providers to monitor the patient's condition in real time. For example, temperature data is randomly generated, heart rate is simulated, and movement data is generated to track the patient's physical activity.

- **Frontend (Web Interface):** The React application provides a user-friendly interface to display the patient's health data and live location. It consists of several pages, such as:

  - **Patient Location:** Displays the current GPS location of the patient on a map using real-time latitude and longitude coordinates.

- **Medical History:** Tracks historical medical data of the patient, including past health records, vitals, and sensor data.
- **Alerts:** Shows any critical health alerts (e.g., abnormal vitals) triggered when sensor data exceeds certain thresholds. For example, an abnormal heart rate or SpO2 level will trigger an alert.
- **Health Monitoring:** Real-time health parameters like temperature, heart rate, ECG, and SpO2 are displayed and updated periodically. This allows healthcare providers to continuously monitor the patient's vital signs.

Diagram of the Architecture

```
+---------------------------------------------------+
|                Smart ICU System                   |
|                                                   |
|  +---------------+        +------------------+    |
|  |  Arduino Mega | <----> |   Cloud (Data)   |    |
|  |  (Simulated   |        |  (Backend/Storage|    |
|  |  Sensors)     |        |   and API)       |    |
|  +---------------+        +------------------+    |
|          |                    |                    |
|          v                    v                    |
|      Frontend (React)      Web Interface           |
|      (Patient Side)      (Location, History, Alerts)|
|      +------------+                              |
|      |  User Interface: React, Google Maps,       |
|      |  Bootstrap for Styling.                   |
|      +------------+                              |
+---------------------------------------------------+
```

4.1.2 Integration with the Backend

In this Smart ICU system, the data from the sensors or simulations is transferred from the **Arduino Mega** to the **cloud** and then displayed on the **frontend** (web interface). This integration can be described in several stages:

Data Collection from Sensors

The Arduino Mega collects data from the simulated sensors (e.g., heart rate, body temperature, GPS location). This data is processed and stored temporarily on the Arduino Mega. In a real-world scenario, this data would be transmitted to the cloud via an IoT platform or network protocol.

Data Transfer to Cloud

In a full implementation, the Arduino could be connected to the cloud via Wi-Fi or GSM modules, sending the data to a cloud server where it is stored in a database. For the purposes of this simulation, the data is transferred to the cloud using simulated API requests. This API can serve as an intermediary between the Arduino and the web interface.

Backend Technologies: This could involve the use of cloud platforms such as **AWS**, **Google Cloud**, or **Microsoft Azure** for storing patient data securely.

Data Retrieval by Frontend

The frontend (React web interface) fetches data from the cloud (or simulated APIs) in real-time. Data such as the patient's **heart rate**, **body temperature**, and **location** are displayed dynamically on the web pages. The **Google Map** component receives the patient's GPS location in real-time, updating the map to reflect the patient's current position.

Alerts System

Based on predefined thresholds for vitals (e.g., abnormal heart rate or body temperature), the system can trigger **alerts** in the frontend. These alerts are fetched from the backend and displayed dynamically on the frontend, notifying healthcare providers of critical conditions.
This architecture ensures that the Smart ICU system works efficiently in monitoring the patient's health remotely, with real-time data visualization and tracking. The integration between the frontend, backend, and sensor simulation ensures that the healthcare providers have immediate access to important health information for timely intervention.

4.2 Data Transmission and Cloud Integration

- Send sensor data wirelessly through http requets or Wi-Fi or Bluetooth modules.
- Store and process the data on a cloud platform (e.g., AWS, Microsoft Azure, MongoDb, or Firebase) for remote access.

4.3 Dashboard Development

- Create a web-based or mobile dashboard to display real-time data for healthcare professionals.
- Dashboard includes:
  - ECG waveform
  - Heart rate graph
  - Temperature graph
  - Body movement indicator
- Implement color-coded indicators (e.g., red for critical, yellow for warning) to highlight abnormal readings.

4.4 Automated Alerts and Notifications

- Integrate a notification system to send instant alerts to medical staff and patients family via SMS, email, or mobile app in emergencies.
- Set predefined threshold values to trigger alerts when vitals deviate from the normal range.

4.5 Predictive Analytics and Machine Learning (Optional)

- Use historical patient data to train machine learning models that can predict health deterioration.
- Implement trend analysis to provide insights and early warnings to medical staff.

4.6 Testing and Validation

- Conduct simulations and real-time tests to validate the accuracy of sensor data and the responsiveness of the alert system.
- Test the system under various scenarios, such as network failure or sensor malfunction, to ensure reliability.

4.7 Security and Data Privacy Implementation

- Apply encryption protocols to protect patient data during transmission and storage.
- Ensure compliance with healthcare standards (like HIPAA or GDPR) to maintain data privacy and confidentiality.

Chapter 5

Requirements

5.1 Hardware Requirements

- **Sensors**
  - 16x2 LCD (I2C)
  - LM35 (Temperature Sensor)
  - MPX10DP (Pressure Sensor)
  - MQ9 MQ135 (Gas Sensors)
  - SPO2 Sensor
  - ECG Sensor
  - Humidity Sensor
  - ADXL (Body Movement Sensor)
  - GPS GSM Modules
  - Buzzer (Alarm)
- **Microcontroller/Development Board**
  - Arduino Mega 2560 or Arduino UNO
- **Communication Modules**
  - Wi-Fi or Bluetooth module (e.g., ESP8266, HC-05)
- **Power Supply**
  - Rechargeable batteries or power adapters for continuous operation
- **Display Unit (Optional)**
  - LCD-12(16x2)or OLED screen for onsite data display
- **Cables and Connectors**
  - For wiring sensors to the microcontroller
- **Backup System**
  - Uninterruptible Power Supply (UPS) to avoid downtime in case of power failure

5.2 Software Requirements

- **Programming Language**
  - Python, C/C++, or JavaScript (for coding the microcontroller and dashboards)
- **Cloud Platform**
  - AWS, Microsoft Azure, Firebase, or IBM Cloud for data storage and remote access
- **Database**
  - SQL or NoSQL databases (e.g., MySQL, MongoDB) for storing patient data
- **Dashboard/Interface Development**
  - Web-based tools like HTML, CSS, and JavaScript for UI
  - Mobile app framework (e.g., React Native, Flutter) for real-time monitoring on smartphones

5.3 Network and Communication Requirements

- **Wi-Fi Connectivity**
  - To transmit data from the microcontroller to the cloud
- **Mobile Network (Optional)**
  - For remote monitoring via mobile devices
- **Router or Hotspot**
  - For uninterrupted data transmission

5.4 Security and Privacy Requirements

- **Encryption Algorithms**
  - AES or RSA encryption for secure data transmission
- **Authentication Mechanism**
  - Username-password or biometric login for authorized access to the dashboard
- **Compliance**
  - HIPAA (Health Insurance Portability and Accountability Act) or GDPR compliance to ensure data privacy

Chapter 6

Components and Technologies

6.1 Overview

The Smart ICU system integrates various technologies to ensure efficient monitoring of the patient's health and location. Below is an overview of the key components and technologies used in the system:

6.1.1 Frontend Technologies

The frontend of the Smart ICU system is responsible for presenting real-time health data and patient location in a user-friendly interface. The following technologies were used to build the frontend:

- **React:** React is the core framework used for building the user interface (UI) of the Smart ICU system. It allows for efficient and dynamic rendering of health data in real time. React components manage the display of various health metrics, patient location, alerts, and historical data. React's declarative approach simplifies the updating of the UI in response to changing data.
- **Bootstrap:** Bootstrap is used to style the UI components, ensuring a responsive and clean design. It provides ready-made components (e.g., buttons, cards, tables) that help in building a professional-looking user interface quickly. Bootstrap's grid system ensures that the layout is responsive across different screen sizes and devices.
- **Google Maps API:** The Google Maps API is used to display the patient's real-time location on an interactive map. The frontend fetches the patient's latitude and longitude data and updates the map, allowing healthcare providers to track the patient's location on the go. Markers and info windows are used to show location details like time and vitals on the map.
- **JavaScript/ES6:** JavaScript and ES6 features (such as arrow functions, promises, and async/await) are utilized to manage dynamic content updates and fetch data from the simulation or backend in real time.

6.1.2 Backend Technologies

The backend of the Smart ICU system manages the storage, processing, and retrieval of patient data. Although the backend was not fully implemented as part of this project, it is assumed that the following technologies would be used in a real-world system:

- **Cloud Storage (e.g., AWS S3, Firebase Storage):** Cloud storage would be used to store historical patient data such as medical records, sensor logs, and health parameters. This would enable healthcare providers to access patient data securely from any location.
- **APIs (e.g., RESTful APIs):** APIs would be used to fetch data from the backend or from external sources (e.g., simulated sensor data, medical history, etc.). The frontend communicates with the backend through API calls to retrieve or update data, ensuring that the system works efficiently and remains scalable.
- **Database (e.g., MySQL, MongoDB):** A database would be used to store structured data such as patient profiles, health monitoring data, and sensor readings. The database ensures that data is properly organized, easy to query, and can be used for analysis and reporting.
- **Node.js (optional):** If a backend server is required to handle requests, Node.js can be used to manage server-side logic, handle incoming requests, and serve the appropriate responses to the frontend. It is well-suited for building RESTful APIs in combination with Express.js.

6.1.3 Simulation

Due to the absence of physical sensors in the current implementation, the system uses a simulation approach to mimic the data that would normally be collected by real hardware. The following sensors were simulated, and their data was fed into the system:

- **Temperature Sensor (e.g., LM35):** The temperature sensor's data is simulated by generating random temperature values within a certain range (e.g., 36°C - 38°C), representing the patient's body temperature.
- **Heart Rate Sensor:** Simulated heart rate data is generated with values typically ranging from 60 bpm to 100 bpm. The data fluctuates periodically to represent normal heart rate changes.
- **ECG Sensor:** The ECG sensor data is simulated with a sinusoidal waveform to mimic the electrical activity of the heart, allowing healthcare providers to monitor the heart's rhythm.
- **SpO2 Sensor (e.g., MAX30100):** Simulated SpO2 levels are generated within a normal range (e.g., 95% to 100%), representing the oxygen saturation in the patient's blood.
- **Air Quality Sensor (e.g., MQ-135):** Random air quality values are generated, representing environmental pollution levels. These values vary periodically to simulate the real-world fluctuations in air quality.

- **Toxic Gas Sensor (e.g., MQ-7, MQ-9):** The sensor data simulates the detection of toxic gases such as carbon monoxide, with values fluctuating to represent exposure levels.
- **Body Movement Sensor (e.g., MPU6050):** The body movement data is simulated by generating random values to track the movement and orientation of the patient, such as walking or lying down.
- **Pressure Sensor (e.g., BMP180):** The pressure sensor data is simulated to reflect changes in atmospheric pressure, influencing the patient's environmental conditions.
- **Humidity Sensor (e.g., DHT11/DHT22):** The sensor generates humidity data within a normal range (e.g., 30%-70%), affecting the comfort of the patient.
- **Alarm System:** The alarm system is simulated to trigger when the data from any sensor exceeds predefined critical thresholds (e.g., high temperature, low SpO2, irregular heart rate). When a threshold is exceeded, an alert is sent to the healthcare provider.
- **GPS Tracking:** The patient's location data (latitude and longitude) is simulated to track the patient's movement. The data is periodically updated to show the current location on the map.

In the simulation, all sensor data is generated dynamically in the frontend using JavaScript, ensuring the system mimics the real-time flow of data that would be produced by physical sensors. This data is sent to the frontend for visualization and analysis.

Chapter 7

Implementation Details

This chapter provides a detailed overview of the implementation of key components in the Smart ICU system. It covers the React code for different pages, sensor simulation, integration of Google Maps for live location tracking, and the implementation of PDF generation for medical history.

7.1 Code Snippets

Below are some key code snippets that illustrate how various pages in the Smart ICU system are implemented.

7.1.1 GPS Tracking Page

The GPS tracking page allows healthcare providers to track the patient's live location using Google Maps. Below is the React code for this page, which updates the patient's location periodically to simulate movement.

```
import React, { useState, useEffect } from "react";
import { GoogleMap, LoadScript, Marker, InfoWindow } from "@react-
   google-maps/api";

const GPSTracking = () => {
  const [patientLocation, setPatientLocation] = useState({ lat:
     37.7749, lng: -122.4194 });
  const [isOpen, setIsOpen] = useState(false);

  useEffect(() => {
    const interval = setInterval(() => {
      // Simulate location change
      setPatientLocation({
        lat: 37.7749 + (Math.random() - 0.5) * 0.01,
        lng: -122.4194 + (Math.random() - 0.5) * 0.01,
      });
    }, 5000);
    return () => clearInterval(interval);
  }, []);

  return (
    <LoadScript googleMapsApiKey="YOUR_GOOGLE_MAPS_API_KEY">
```

```
      <GoogleMap center={patientLocation} zoom={15} mapContainerStyle={{ height: "400px", width: "100%" }}>
        <Marker position={patientLocation} onClick={() => setIsOpen(!isOpen)} />
        {isOpen && (
          <InfoWindow position={patientLocation} onCloseClick={() => setIsOpen(false)}>
            <div>Patient Location</div>
          </InfoWindow>
        )}
      </GoogleMap>
    </LoadScript>
  );
};

export default GPSTracking;
```

This code initializes the Google Map and periodically updates the patient's location with random values to simulate movement every 5 seconds.

7.1.2 Medical History Page

The medical history page displays the patient's health data over time. Below is the React code to render the medical history in a table format.

```
import React from 'react';

const MedicalHistory = ({ historyData }) => {
  return (
    <div>
      <h2>Patient's Medical History</h2>
      <table className="table">
        <thead>
          <tr>
            <th>Date</th>
            <th>Health Parameter</th>
            <th>Value</th>
          </tr>
        </thead>
        <tbody>
          {historyData.map((entry, index) => (
            <tr key={index}>
              <td>{entry.date}</td>
              <td>{entry.parameter}</td>
              <td>{entry.value}</td>
            </tr>
          ))}
        </tbody>
      </table>
    </div>
  );
};

export default MedicalHistory;
```

This component takes ‘historyData‘ as a prop, which contains the patient's medical records, and displays them in a table.

7.1.3 Alert System

The alert system monitors critical health parameters such as temperature and heart rate. If any parameter falls outside the normal range, an alert is displayed to notify healthcare providers. Below is the React code for the alert system.

```
import React, { useState, useEffect } from 'react';

const AlertSystem = ({ healthData }) => {
  const [alert, setAlert] = useState(null);

  useEffect(() => {
    if (healthData.temperature > 38) {
      setAlert('High Temperature! Immediate attention required.');
    } else if (healthData.heartRate < 60) {
      setAlert('Low Heart Rate! Immediate attention required.');
    } else {
      setAlert(null);
    }
  }, [healthData]);

  return (
    <div>
      {alert && (
        <div className="alert alert-danger">
          <strong>Alert:</strong> {alert}
        </div>
      )}
    </div>
  );
};

export default AlertSystem;
```

This code triggers alerts based on health data changes, displaying an alert when critical values like high temperature or low heart rate are detected.

7.2 Sensor Simulation

In the absence of physical sensors, we simulate sensor data to mirror the real-world behavior of various health monitoring sensors. These simulated sensors include:

- **Temperature Sensor (LM35):** Random temperature values between 36°C and 38°C simulate the patient's body temperature.
- **ECG Sensor:** Simulated ECG signals that represent the patient's heart rhythm, typically generated using sinusoidal waves or random heart rate values.
- **Heart Rate Sensor:** Simulated heart rate values ranging from 60 bpm to 100 bpm, updated periodically.
- **SpO2 Sensor:** Simulated oxygen saturation levels ranging between 95% and 100%.
- **Other Sensors:** Other health-related metrics like body movement, pressure, and humidity are also simulated using predefined ranges and periodic updates.

These simulated values allow the frontend to reflect real-time monitoring, providing an interactive experience for healthcare providers.

7.3 Integration of Google Maps

The Smart ICU system tracks the patient's live location using the Google Maps API. The patient's location is represented by latitude and longitude coordinates, which are updated periodically. The map is rendered on the frontend with a marker that moves to reflect the patient's current location.
Here's a code snippet that demonstrates how the patient's location is updated on Google Maps:

```
import React, { useState, useEffect } from 'react';
import { GoogleMap, LoadScript, Marker } from '@react-google-maps/api';

const GPSTracking = () => {
  const [patientLocation, setPatientLocation] = useState({ lat:
     37.7749, lng: -122.4194 });

  useEffect(() => {
    const interval = setInterval(() => {
      // Simulate location update
      setPatientLocation({
        lat: 37.7749 + (Math.random() - 0.5) * 0.01,
        lng: -122.4194 + (Math.random() - 0.5) * 0.01,
      });
    }, 5000);
    return () => clearInterval(interval);
  }, []);

  return (
    <LoadScript googleMapsApiKey="YOUR_GOOGLE_MAPS_API_KEY">
      <GoogleMap
        center={patientLocation}
        zoom={15}
        mapContainerStyle={{ height: "400px", width: "100%" }}
      >
        <Marker position={patientLocation} />
      </GoogleMap>
    </LoadScript>
  );
};
```

This code initializes the map and updates the patient's position every 5 seconds by modifying the ‘patientLocation‘ state.

7.4 PDF Generation

The Smart ICU system also includes functionality to download the patient's medical history as a PDF document. The following snippet demonstrates how this feature is implemented using the 'jsPDF' library.

```
import { jsPDF } from "jspdf";

const downloadPDF = (historyData) => {
  const doc = new jsPDF();

  doc.text("Patient Medical History", 20, 20);
  historyData.forEach((entry, index) => {
    doc.text(`Date: ${entry.date}`, 20, 30 + index * 10);
    doc.text(`Parameter: ${entry.parameter}`, 20, 40 + index * 10);
    doc.text(`Value: ${entry.value}`, 20, 50 + index * 10);
  });

  doc.save("medical_history.pdf");
};

// Example button to trigger PDF download
<button onClick={() => downloadPDF(historyData)}>Download PDF</button>
```

The code uses 'jsPDF' to generate a PDF document with the patient's medical data, and provides a button to download the file as "medical$_history.pdf$."

Chapter 8

Results and Discussion

8.1 Screenshots/Visuals

In this section, we provide screenshots of the key components of the system, including the patient-side user interface (UI), the dashboard, the GPS tracking page, the alert system, and the medical history page. These visuals demonstrate the system's ability to track real-time patient data and display it in a user-friendly interface. The following screenshots capture these functionalities:

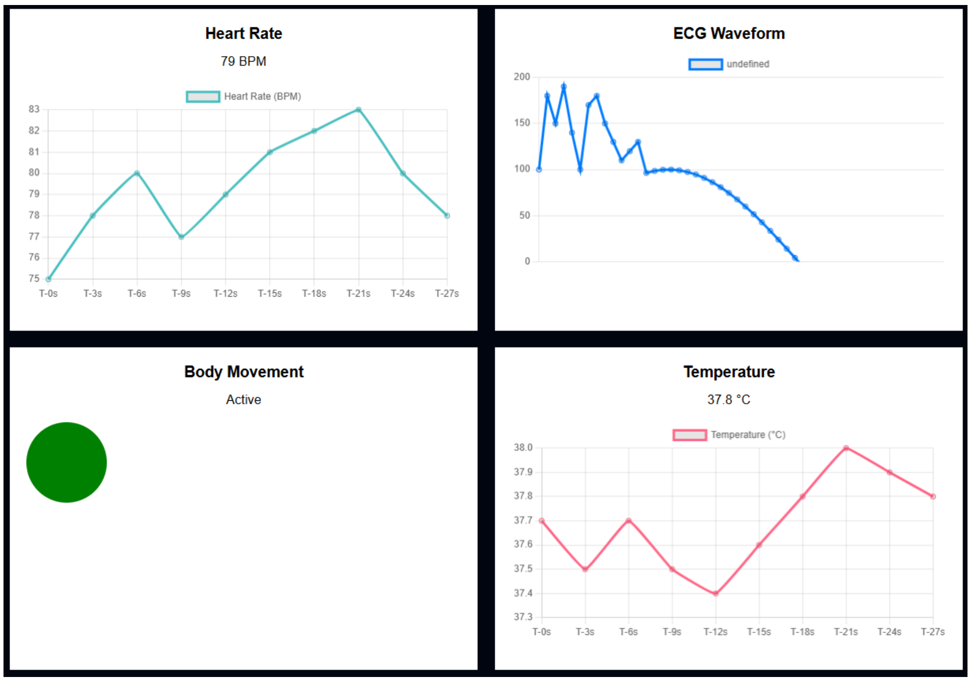


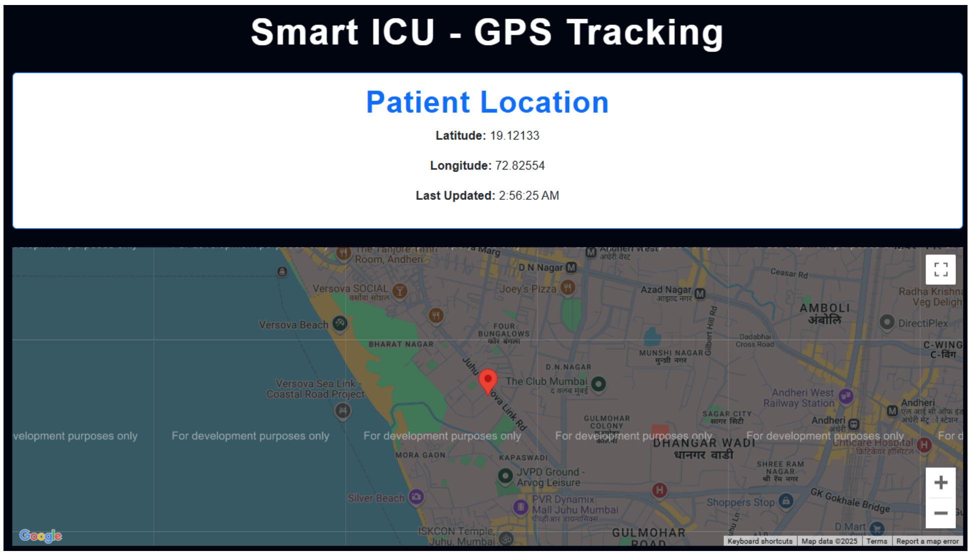
Figure 8.1: Patient-Side Dashboard displaying Health Metrics and Patient Status

8.2 Testing and Validation

The system underwent thorough testing to ensure its functionality and accuracy in real-world scenarios. Below are the major areas of testing and validation:

8.2.1 Patient Location Tracking

The GPS tracking feature was extensively tested by simulating different movements for the patient's location. We observed that the system accurately displayed the patient's location on the Google Map, with updates occurring at regular intervals. The simulated movements (latitude and longitude updates) were tested in different regions to ensure that the real-time tracking was functional and responsive.
Location tracking is a critical aspect of the Smart ICU system, especially for patients who may be mobile or at risk of wandering. Real-time tracking of a patient's location provides healthcare providers with valuable data on the patient's whereabouts at all times, enhancing patient safety. For example, if a patient with critical health conditions moves unexpectedly or leaves a designated area, the system can alert medical staff and provide immediate assistance.
Moreover, location tracking can be particularly useful in emergency situations. In cases where a patient's condition deteriorates rapidly, knowing their exact location allows for quicker intervention, potentially saving lives. It also helps in managing patients in large healthcare facilities or even remotely monitoring patients at home, ensuring that healthcare professionals can always provide timely support and care.
By using GPS for real-time tracking, the system can improve overall patient care, facilitate quicker responses in emergencies, and ensure that healthcare providers are always aware of the patient's location, especially in critical or unpredictable situations.



Smart ICU - GPS Tracking

Patient Location

Latitude: 19.12133

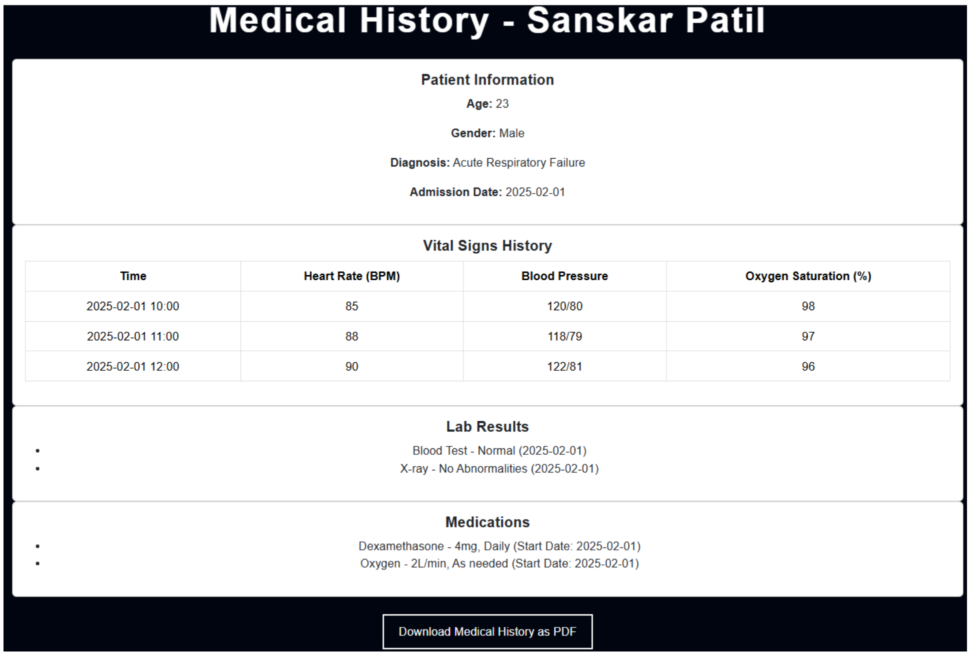
Longitude: 72.82554

Last Updated: 2:56:25 AM

Figure 8.2: GPS Tracking Page showing Patient's Live Location on Google Maps

8.2.2 Medical History Management

We tested the medical history management feature by simulating the entry and display of patient health data, such as heart rate, body temperature, and oxygen saturation (SpO2) levels. The medical history page effectively stored and presented these data points, showing a continuous record of the patient's health over time. Additionally, the option to download the medical history as a PDF was verified to ensure the functionality was seamless.
Medical history management is a crucial component of any healthcare system, especially in the context of continuous patient monitoring. By tracking a patient's health data over time, healthcare providers gain insights into trends, helping them to better understand the patient's condition and make more informed decisions. For example, historical data on body temperature, heart rate, and oxygen levels can indicate whether a patient's condition is improving or deteriorating, which is vital for timely interventions.
Additionally, having a centralized system to store and view patient data allows for better coordination among healthcare professionals. This is especially important in situations where multiple caregivers are involved, as it ensures that everyone has access to the most up-to-date information. Furthermore, the ability to download the medical history as a PDF provides an easy way to share the patient's records with other medical professionals or keep them for personal reference.
By providing both real-time data and historical context, the medical history feature ensures that healthcare providers can make decisions based not only on current health readings but also on past trends, contributing to more personalized and effective care.



Medical History - Sanskar Patil

Patient Information

Age: 23

Gender: Male

Diagnosis: Acute Respiratory Failure

Admission Date: 2025-02-01

Vital Signs History

Time	Heart Rate (BPM)	Blood Pressure	Oxygen Saturation (%)
2025-02-01 10:00	85	120/80	98
2025-02-01 11:00	88	118/79	97
2025-02-01 12:00	90	122/81	96

Lab Results

- Blood Test - Normal (2025-02-01)
- X-ray - No Abnormalities (2025-02-01)

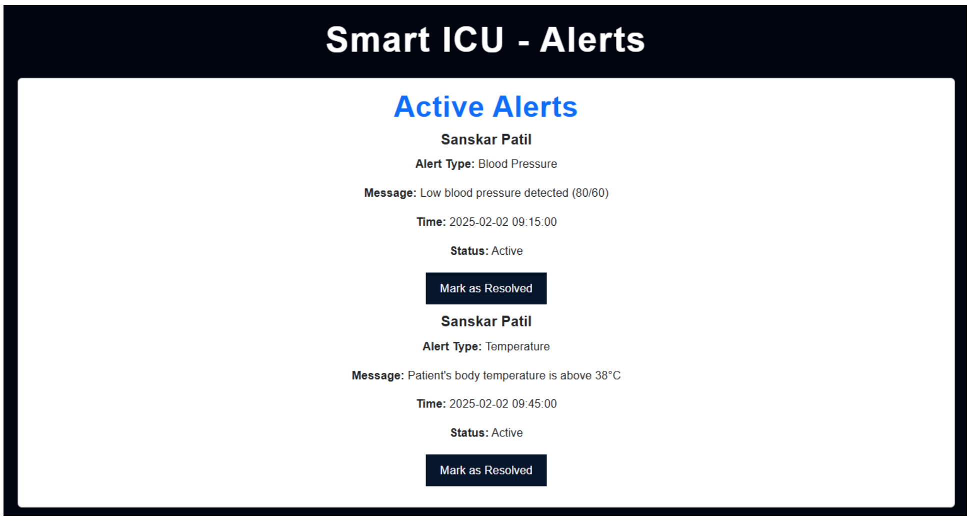
Medications

- Dexamethasone - 4mg, Daily (Start Date: 2025-02-01)
- Oxygen - 2L/min, As needed (Start Date: 2025-02-01)

Figure 8.3: Medical History Page displaying Historical Health Data and Patient's Health Profile

8.2.3 Alert System

The alert system was tested by simulating abnormal sensor readings. For instance, we simulated high body temperature and abnormal heart rate values, and the system successfully triggered alerts accordingly. These alerts were displayed on the frontend, notifying healthcare providers about the critical condition of the patient.
The alert system plays a crucial role in ensuring timely medical interventions, particularly in cases where a patient's condition deteriorates rapidly. By continuously monitoring vital signs such as heart rate, oxygen levels, and body temperature, the system can detect anomalies and immediately notify caregivers. This feature is especially important for patients with chronic illnesses, elderly individuals, or those in critical care units.
In real-world scenarios, delays in responding to critical health changes can lead to severe complications or even fatalities. The alert mechanism minimizes this risk by providing instant notifications through visual and auditory signals. The system is designed to trigger alerts based on predefined threshold values. For example, if the heart rate drops below or exceeds a safe range, or if oxygen saturation falls below critical levels, an immediate warning is issued.
Furthermore, integrating the alert system with a digital healthcare platform ensures that notifications can be sent not only via the patient-side dashboard but also through mobile notifications, emails, or direct alerts to medical personnel. This guarantees that healthcare providers can respond promptly, even if they are not physically present with the patient.
By automating the detection of abnormal health conditions and facilitating real-time responses, the alert system significantly enhances patient safety and reduces the burden on healthcare staff.


Smart ICU - Alerts

Active Alerts

Sanskar Patil

Alert Type: Blood Pressure

Message: Low blood pressure detected (80/60)

Time: 2025-02-02 09:15:00

Status: Active

Sanskar Patil

Alert Type: Temperature

Message: Patient's body temperature is above 38°C

Time: 2025-02-02 09:45:00

Status: Active

Figure 8.4: Alert System Page showing Critical Health Alerts such as Abnormal Temperature or Heart Rate

8.3 Challenges and Solutions

Throughout the development process, we faced a few challenges, which were resolved as follows:

8.3.1 Google Maps Integration

One of the main challenges was integrating Google Maps into the frontend to display the patient's live location. Initially, we faced issues related to API key configuration and display responsiveness. This was resolved by ensuring the correct API key was used and setting proper parameters for rendering the map. The use of '@react-google-maps/api' helped streamline the process and provide easy-to-implement mapping features.

8.3.2 Managing Simulation Data

Since we simulated sensor data for the prototype, managing dynamic and realistic data for multiple sensors such as temperature, heart rate, SpO2, and others posed a challenge. We overcame this by writing JavaScript code that simulated sensor readings and updated them at regular intervals, mimicking real-world data flow.

8.3.3 User Interface and Responsiveness

Ensuring that the patient-side user interface (UI) was both intuitive and responsive across different devices was another challenge. To address this, we utilized Bootstrap for responsive design and ensured that the UI elements were correctly aligned and adjusted according to screen size. This allowed us to maintain a clean and functional UI on both desktop and mobile devices.

Chapter 9

Future Work and Enhancements

9.1 Potential Improvements

While the current system effectively simulates a smart patient monitoring environment, several enhancements can be made to improve its accuracy, functionality, and real-world applicability. Some of the key areas for improvement include:

9.1.1 Integration of Real Sensors

Currently, the system relies on simulated sensor data to mimic real-world conditions. A significant enhancement would be replacing simulated data with real sensor inputs. By integrating actual medical sensors such as:

- Real-time ECG monitors (e.g., AD8232) for precise heart rate tracking.
- Pulse oximeters (e.g., MAX30100) for accurate SpO measurements.
- Blood pressure sensors to provide continuous BP monitoring.
- Wearable accelerometers (e.g., MPU6050) for detecting patient movement and fall detection.

This would allow for more reliable data collection and better accuracy in patient health monitoring.

9.1.2 Advanced Alert System with AI Integration

The current alert system is based on threshold values for different health parameters. Future enhancements can incorporate Artificial Intelligence (AI) and Machine Learning (ML) to:

- Detect patterns in a patient's health data over time.
- Predict potential health risks based on historical trends.
- Reduce false alarms by distinguishing between critical conditions and temporary fluctuations.

AI-driven alerts would make the system more intelligent and capable of providing early warnings for potential health risks.

9.1.3 Cloud-Based Data Storage and Remote Access

At present, patient data is managed on the frontend. A cloud-based infrastructure can be implemented to:

- Store real-time patient data securely on cloud servers.
- Enable healthcare providers to access patient history from any location.
- Allow for data analysis and trend visualization over extended periods.

Integration with platforms like Firebase, AWS IoT, or Google Cloud would enhance scalability and data accessibility.

9.1.4 Mobile App Integration

To improve accessibility, a mobile application can be developed alongside the web interface. This would allow:

- Patients and caregivers to receive real-time alerts and notifications.
- Doctors to monitor patient vitals remotely from their smartphones.
- Location tracking via GPS to be more seamless and efficient.

9.1.5 Scalability for Large-Scale Implementation

For real-world deployment in hospitals or remote healthcare facilities, the system should be scalable to support:

- Multiple patients being monitored simultaneously.
- Secure multi-user authentication for different levels of access (e.g., doctors, nurses, family members).
- Integration with hospital databases and Electronic Health Records (EHRs).

Ensuring scalability would make the system applicable in both home-based healthcare and hospital environments.

9.1.6 IoT and Wearable Device Connectivity

Future enhancements could involve connecting the system with wearable healthcare devices like:

- Smartwatches with heart rate and SpO monitoring.
- IoT-enabled glucose monitors for diabetic patients.
- Bluetooth-enabled blood pressure cuffs.

These integrations would provide continuous health tracking without requiring the patient to be physically wired to monitoring devices.

Chapter 10

Conclusion

The Smart ICU System overcomes the limitations of traditional critical care by offering continuous, real-time, and patient-specific monitoring. By integrating IoT sensors, cloud computing, and mobile communications, the system enables instant alerts and remote access, allowing physicians to respond swiftly to emergencies and identify potential health risks before they escalate.
The scalability of this design ensures it can be adapted to different hospital environments, making it a cost-effective and efficient solution. By reducing the burden on medical staff and enhancing patient safety, the Smart ICU system improves healthcare outcomes.
Our advanced health monitoring system incorporates multiple sensors, including LM35 temperature sensors, MEMS accelerometers, SPO2 sensors, ECG modules, gas sensors, and motion detectors. These components facilitate the real-time tracking of vital health indicators such as temperature, oxygen levels, heart activity, air quality, and movement, ensuring proactive and timely medical intervention.
Additionally, this solution provides healthcare professionals with secure, direct access to critical patient data, enabling them to make informed decisions swiftly. Despite challenges such as data security and integration complexities, our Smart ICU system enhances patient care, optimizes medical workflows, and elevates healthcare standards, ultimately leading to better patient outcomes and hospital efficiency.

References

1. E. Al Alkeem et al., ”New secure healthcare system using cloud of things,” Springer Science+Business Media New York, 2017.
   Available at: `https://www.researchgate.net/publication/316749947_New_secure_healthcare_system_using_cloud_of_things`

2. Y. Kim, S. Lee, and S. Lee, ”Coexistence of ZigBee-based WBAN and WiFi for Health Telemonitoring Systems,” IEEE Journal of Biomedical and Health Informatics, DOI: 10.1109/JBHI.2014.2387867.

3. M. M. Baig and H. Gholamhosseini, ”Smart Health Monitoring Systems: An Overview of Design and Modeling,” Springer Science+Business Media New York, 2013.

4. S. M. Riazulislam et al., ”The Internet of Things for Health Care: A Comprehensive Survey,” IEEE Transactions, DOI: 10.1109/TDSC.2015.2406699.
   Available at: `https://www.semanticscholar.org/paper/The-Internet-of-Things-for-Health-Care%3AA-Survey-Islam-Kwak/cddb22908f28a1636cbbdeb3a4f0e00f9cef05a9`

5. A. Mdhaffar et al., ”IoT-based Health Monitoring via LoRaWAN,” IEEE EUROCON 2017.
   Available at: `https://www.researchgate.net/publication/319169748_IoT-based_health_monitoring_via_LoRaWAN`

6. M. M. Masud et al., ”Resource-Aware Mobile-Based Health Monitoring,” 2015 IEEE.

7. A. Bansal et al., ”Remote health monitoring system for detecting cardiac disorders,” IET Syst.Biol., vol. 9, no. 6, pp. 309–314, 2015.

8. H. Al-Hamadi and I. Chen, ”Trust-Based Decision Making for Health IoT Systems,” IEEE Internet of Things Journal, DOI: 10.1109/JIOT.2017.2736446.